Date: 2/6/04 Express Mail Label No. EV214950045US

Inventors: Alia Karin Atlas, Raveendra Torvi

Attorney's Docket No.: 2390.2009-001

RAPID ALTERNATE PATHS FOR NETWORK DESTINATIONS

RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/508,766, filed October 3, 2003. The entire teachings of the above application are incorporated herein by reference. The entire teachings of U.S. Provisional Application "Network Routing Algorithm" by Alia Karin Atlas and Raveendra Torvi, filed on February 6, 2004, are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Networks are used to transmit data (also called network traffic) between network devices connected by links. A path used for transmission of data between two network devices may go through several intermediate network devices and links. A network device may be a router, a computer, a processor in a multiprocessor computer, or any other device as long as it is capable of performing the required network tasks.

A network in general has multiple paths between a given source and a given destination, uses some routing protocol to select a set of the available paths and should be capable of tolerating some links between network devices breaking. One example of such a network is a packet-switched network transmitting packets consisting of binary data. The packets are sent from one network device (also called a host or a node) to another network device, usually through several intermediate network devices known as routers that determine the next hop, i.e. the next network device in the path to the destination.

The transmission of data between network devices, such as routers, may be accomplished using a variety of technologies: wires, phone lines, optical links, wireless links, and others. The low-level protocols used by implementations of these technologies are known as physical layer (or level 1) protocols.

Internet Protocol (IP) based networks, i.e., networks conforming to Request for Comments (RFC) 0791 and RFC 1349 distributed by the Internet Engineering Task Force (IETF) are a popular type of packet-switched networks. The IETF develops, distributes, and maintains a variety of network standards commonly referred to by their numbers as RFCs. A global IP network comprising a large number of interconnected local networks is known as the Internet. A full set of RFC's is available at the IETF's Internet site.

IP is an example of a network layer (or level 3) protocol. Network layer protocols rely on link layer (or level 2) protocols. These protocols may also involve routing packets. A popular type of link layer protocol is Ethernet.

A host is connected to at least one other host. A network router is a host connected to two or more other hosts. Hereinafter, these other hosts will be called the hosts's neighbors and the connection ports on the hosts will be called interfaces. A transmission over a single link between two neighbors is called a "hop". Upon receiving a packet, a router decides the best, for some definition of best, next hop to use, such that the packet eventually arrives at its destination. This decision is usually made using the information carried within the packet, such as the packet's destination host. A router may also use information about the network's topology, i.e. how other hosts are interconnected to determine which interface to direct an incoming packet. Another variable that the router may consider in making its decision is the input interface on which the packet has arrived. Alternatively, a packet may simply carry a label, which the router would use to determine the output interface by using a table per input interface indexed by the labels and containing the output interfaces. One such label-based forwarding mechanism is known as Multi-Protocol Label Switching (MPLS, RFC 3031).

Normally, routing selects the lowest cost path to the destination. Given a network topology, each link in each direction is assigned a positive number, its "cost". Given this information, the path chosen for forwarding a packet is the route with the least sum of costs. Each router stores in its memory the network topology and the relevant link costs and for each destination D determines to which neighbor it sends a D-addressed packet so that it travels the lowest cost path. However, other considerations may apply, such as reducing strain on some links in the network, or prioritizing delivery of some packets over others.

After a change in network topology (for example, when a link between two hosts is broken), the routing decisions made by routers may also change. In other words, a packet arriving to a router from interface A may be forwarded to interface B before the change and to interface C after the change. To minimize network traffic loss after a topology change, it is preferable that even before network routers become aware of what the new topology is and make proper adjustments in their traffic forwarding strategy, they continue to forward packets to their destinations. In other words, it is preferable that the forwarding strategy does not cause a loss of traffic in case of topology change even when forwarding decisions are based on stale (pre-change) topology and before even the occurrence of the change becomes known.

Obviously, this task is simply impossible in some cases (to take an extreme example, when all links in a network fail) and trivial in others (for example, when an extra link or links between routers are added to a network without other changes, the old forwarding strategy would remain functional by simply ignoring the added links). However, usually links between routers fail one at a time and networks have enough link redundancy to allow traffic redirection. Many routing algorithms, such as OSPF, rely on link state advertisements (LSAs) to inform all the other routers in a routing group of link status. In the case of a down link, the routers directly connected to that link (that are still able to function) broadcast (flood) a new LSA indicating the down link. These routers compute routes based on the new information and install those new routes. As the other routers in the routing group receive the LSA, they also recompute routes based on the new information and install those new routes as well.

The goal is for all of the routers to compute the same routes. Once their view of the network from the LSAs becomes consistent, the routes become identical since they use the same algorithm to compute the routes. However, since the information takes a different amount of time to get to each router, each router may take a different amount of time to compute the new routes, and there may be more than one link event at any given time, different routers may have different routes installed for some time after a topology change. Inconsistent routes may cause routing loops (closed-loops), where traffic is forwarded in a ring of routers, never to reach their final destination.

SUMMARY OF THE INVENTION

This invention provides a method of forwarding network traffic comprising storing a link to an alternate neighbor node at a network node, and upon detecting network traffic coming from a primary neighbor node at the network node, the primary neighbor node being primary for the network node with respect to the network traffic, forwarding the network traffic to the alternate neighbor node.

The alternate neighbor node may be loop-free for the network traffic with respect to the primary neighbor node.

The alternate neighbor node may be loop-free for the network traffic with respect to a primary neighbor of the primary neighbor node with respect to the network traffic.

The alternate neighbor node may be loop-free for the network traffic with respect to all primary neighbors of the primary neighbor node with respect to the network traffic.

The alternate neighbor node may be loop-free for the network traffic with respect to all nodes failing together with primary neighbors of the primary neighbor node with respect to the network traffic.

The alternate neighbor node may be the first node on a path to the traffic destination and that path does not utilize any links or nodes that are known to potentially fail simultaneously with any of the primary neighbor's primary next hops.

The alternate neighbor node may be the first node on a path to the traffic destination and that path does not utilize any links or nodes that are known to potentially fail simultaneously with any of the primary neighbor's primary neighbors.

The alternate neighbor node may be loop-free for the network traffic with respect to a primary remote node, the primary remote node being a node in a sequence of nodes starting at the network node, each node in the sequence being a primary neighbor of the prior node in the sequence with respect to the network traffic and each node in the sequence being an alternate neighbor node of the latter node in the sequence.

The alternate neighbor node may be loop-free for the network traffic with respect to a primary remote node, the primary remote node being a node in a sequence of nodes starting at the network node, each node in the sequence being a primary neighbor of the prior node in the sequence with respect to the network traffic.

The alternate neighbor node may be loop-free for the network traffic with respect to a primary neighbor of the remote primary node with respect to the network traffic.

The alternate neighbor node may be loop-free for the network traffic with respect to all primary neighbors of the remote primary node with respect to the network traffic.

The alternate neighbor node may be loop-free for the network traffic with respect to all nodes failing together with primary neighbors of the remote primary node with respect to the network traffic.

The network node may be primary for the alternate neighbor node with respect to the network traffic.

The alternate neighbor node may depend on the network traffic's destination.

The alternate neighbor node may be determined using a modified Dijkstra algorithm.

The network packet may comprise data packets.

This invention also provides a method of forwarding network traffic to a globally defined destination comprising: at a network node, for the destination, storing a function mapping inputs of the network node to outputs of the network node; and forwarding the network traffic arriving to the network node from the inputs to the outputs according to the mapping function.

This invention also provides a method of forwarding network traffic to a globally defined destination without multicasting the network traffic comprising: at a network node, for the destination, storing a function mapping inputs of the network node to outputs of the network node; and forwarding the network traffic arriving to the network node from the inputs to the outputs according to the mapping function.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1 is a schematic illustration of one embodiment of this invention.

Fig. 2 illustrates the functioning of the Dijkstra algorithm.

Fig. 3 shows the functioning of a loop-free alternate node.

Fig. 4 shows the functioning of a U-turn alternate node.

Fig. 5 illustrates application of the Dijkstra algorithm for some embodiments of this invention.

Fig. 6 shows Basic Topology with Primary and Alternate

Fig. 7 shows Example Topology with Local SRLGs

Fig. 8 shows Topology for Terminology

Fig. 9 shows Topology for Terminology of Looping Neighbors

Fig. 10 shows U-Turn Neighbor with ECMP

Fig. 11 shows ECMP Neighbor Which Is Not a ECMP U-Turn Neighbor

Fig. 12 shows U-Turn Alternate Example

Fig. 13 shows Topology for Loop-Free Proof

Fig. 14 shows New Forwarding Rule for IP/LDP

Fig. 15 shows Forwarding Rules for Traffic From Primary Next-Hop

Fig. 16 shows Example Forwarding Tables

Fig. 17 shows Broadcast Interface Translated to Pseudo-Node

Fig. 18 shows Converging to a Loop-Free Neighbor

Fig. 19 shows Converging to a U-Turn Neighbor

Fig. 20 shows Using a U-Turn Alternate and Converging to a U-Turn Neighbor

Fig. 21 shows Example Where 2 U-Turn Alternates lead to Forwarding Loop

Fig. 22 shows Timeline of traffic loss after link failure

Fig. 23 shows Eligible Alternate Link Capability Dependence

Fig. 24 shows Eligible U-Turn Recipient Link Capability Dependence

Fig. 25 shows OSPF Inter-Area Alternate Inheritance

DETAILED DESCRIPTION OF THE INVENTION

A description of preferred embodiments of the invention follows.

Rather than rely only on the general routing algorithm and waiting for all nodes to converge on a uniform set of routes, embodiments of this invention provide mechanisms to properly route packets even before the new routes are calculated. The embodiments use alternate routes when the primary routes (i.e. routes calculated before the link is down) traverse a down link. The only routers switching over to alternate routes are the routers connected to the down link. The alternate paths may be used until the rest of the network converges.

Each node may compute the output link for each destination using an algorithm, for example, the well-know Dijkstra's algorithm. In the event of a failure, the nodes are made aware of the failure. Then the algorithm must be run again, but before all nodes run their algorithms (i.e. before the network converges), this invention permits the network to remain functional. To allow this, during the initial computation for each node, embodiments of this invention define, if possible, at least one link to an alternate neighbor to which packets are forwarded in case of a link failure. The alternate link from a node S is preferably to a node with a path to the destination bypassing S. The alternate neighbor may have as its primary path to the destination a path flowing through S which would result in a loop. To avoid this, some embodiments of this invention allow for U-turn. In the case of a U-turn, the alternate neighbor follows a rule by which when the alternate neighbor N receives a packet from a node S to which in would normally send the packet, it sends it to an alternate node such that the packet does not pass through S or N again. The U-turns may be cascaded.

Hereinafter the functioning of embodiments of this invention will be described with attention to single link failures, but it must be understood that the invention's embodiments remain functional in many cases of multiple link failures, host failures, and other topology changes as well.

Fig. 1 shows an embodiment of this invention including nodes 10, 11, 13-17 and the rest of the network 18. When a packet destined for the node 17 arrives to the node 10 from the node 13, the node 10 decides whether to send this packet to the node 11, 14, 15, or 16. 10's decision may be made by choosing a path on the basis of the IP address of the packet's destination or on the basis of the MPLS label attached to the packet. The packets having the same MPLS label are said to belong to the same Forwarding Equivalence Class (FEC). Each direction of each link in the network 21 is assigned a non-negative cost. The paths to choose are calculated for each IP or each MPLS by choosing the path with the least sum of costs of links for each destination and are stored at each node.

In Fig. 1, when a packet 19 arrives from the node 13 to node 10 via link 20, it is normally routed (based, for example on its IP destination or FEC) to the node 11 via link 12. When the node 10 determines that the link 12 is broken, as shown in Fig. 1, it decides whether to forward the packet to any of the nodes 13-16 assuming that they are not aware that the link 12 is broken. This decision may be different from the decision made after the entire network 21 becomes aware of this topology change and computes the new paths based on this change. After the new paths are computed, all the nodes in the network are choosing the best strategy for packet forwarding and are guaranteed not to send a packet through a closed loop. This assumption may not be made before the new paths are computed by each node and thus the node 10 chooses the alternate node for the packet 19 so that the packet 19 does not end up on a closed loop route.

For each packet's FEC or IP destination, a node may be determined as an alternate mode based on whether each of its neighbors belongs to one of the following classes: (a) loop-free alternate neighbors that do not forward through the primary next node 11; (b) loop-free alternate neighbors that do forward through the primary next node 11; or (c) U-turn neighbors with a loop-free alternate neighbor node. Within each category a priority may be assigned to each neighbor within the category. When the link through which a packet is normally forwarded is broken, this packet is forwarded to the neighbor in the category (a), if any available, which has the highest priority in the category (a), otherwise, to the neighbor in the category (b), if any available, which has the highest priority in the category (b), otherwise, to the neighbor in the category (c), if any available, which has the highest priority in the category (c). The node 10 decides by this method to which of the nodes 13-16 to forward the packet 19 after the link 12 is broken.

The first two categories under consideration are (a) loop-free alternate neighbors that do not forward through the primary next node and (b) loop-free alternate neighbors that do forward through the primary next node. A loop-free neighbor for a node for a given destination is a neighbor which does not forward a packet to the destination through the node. In other words, the node 16 (see Fig. 1) is loop-free with respect to the node 10 and the packet 19 if, after receiving the packet 19, the node 16 normally does not forward it to any route that may eventually lead back to the node 10 including through link 26. The loop-free neighbors that normally do not forward the packet 19 to the route going through the node 11 (the node to which the packet 19 would normally go from the node 10 were the link 12 not broken) belong to the category (a), the other loop-free neighbors belong to the category (b). In this paragraph, "normally" means using a least-cost algorithm (such as Dijkstra's) without knowing that the link 12 is down.

A neighbor belonging to the category (c), U-turn neighbors with a loop-free alternate neighbor node, for a node for a given destination is a neighbor that normally forwards a packet to the destination back to the node and also has at least one neighbor which forwards the packet through a path going neither through the node nor through the node's neighbor. The node 14 belongs to the category (c) with respect to the node 10 and the packet 19 if it would normally forward the packet 19 to the node 10 via the link 24 and if it has a neighbor 22 which would normally forward the packet 19 neither through the node 14 nor through the node 10. In this paragraph, "normally" means using a least-cost algorithm (such as Dijkstra's) assuming the link 12 is up.

In embodiments of this invention utilizing category (c) neighbors after a link failure, such neighbors do not forward all the incoming packets using a least-cost algorithm; otherwise the packets arriving from the node 10 to node 14 via link 24 would be forwarded back to the node 10 via the same link 24 in the opposite direction, creating a loop. Whenever a category (c) neighbor 14 receives a packet 19 from a node 10, it or new routing table for link 24 in node 14 checks whether the stored primary path requires it to forward packet 19 back to the node 10. If so, it forwards the packet 19 to an alternate neighbor node 22 which guarantees that the packet on its way to its destination does not pass again through either node 10 or node 14. In other words, the node 14 stores for the combination to node 17, from node 10 a neighbor 22 where it sends packets destined for the node 17 when they arrive via link 24.

As mentioned above, the routing through the category (a), (b), or (c) neighbors continues only until the nodes in the network become aware of the changed topology (for example, by LSA) and switch to the newly computed routes. The nodes directly connected to the down link must wait a configurable hold-down time before switching to the new routes to give the rest of the network time to converge.

Functioning of one embodiment is illustrated in Fig. 3. The solid arrows indicate the routing of packets addressed to link 304 before the link 306 is broken, in particular, 301-303-304 and 302-304 (not passing through 301). After the link 306 is broken, instead of 301-303-304, the packets travel along the route 301-302-304. This alternative routing by the node 301 is shown by a dashed line. The node 302 is a category (a) neighbor for the node 301 and destination 304. When node 301 sees a break at 306, it forwards packets destined to the node 304 on the alternate link 307. The node 302 forwards the packet toward the node 304 via the same link 308, on which it would normally forward packets received from links 309 and 310 and destined to the node 304.

Functioning of another embodiment is shown in Fig. 4. The solid arrows indicate the routing of packets addressed to the node 404 before the link 406 is broken, in particular, 401-403-404, 402-401-403-404, and 405-404 (not passing through 401 or 402). After the link 406 is broken, instead of 401-403-404, the packets travel along the route 401-402-405-404 and instead of 402-401-403-404, the packets travel along the route 402-401-402-405-404, thus making a U-turn at the node 401. This alternative routing by the nodes 401 and 402 is shown by dashed lines. The node 402 is a category (c) neighbor for the node 401 and destination 404. The node 401 uses the alternate link 406. The node 402 recognizes that it has received a packet from a link onto which it normally would send packets with such destination and thus uses its alternate link 407. The node 405 forwards the packet via its normal primary link 408.

Note that some embodiment of this invention may employ only neighbors of type (a), and/or neighbors of type (b), and/or neighbors of type (c) for the temporary routing. Note also that when only neighbors of type (a) and/or type (b) are used, no behavior modification is required for the nodes not connected with the broken link. Incorporation of all these types into an embodiment of this invention provides more opportunities for determining appropriate alternates.

The following equations determine whether a node's neighbor is a loop-free neighbor for a given destination.

If a node in the network is loop-free with respect to the source S for a particular destination D, the following equation is true. If it is true, then the particular node N is loop-free with respect to the source for that destination.

$$\text{Distance}_{S}(N, D) < \text{Distance}_{opt}(N, S) + \text{Distance}_{opt}(S, D)$$

If the above equation is true, then N does not forward traffic to S in order to reach D because N has a shorter path. The $\text{Distance}_{S}(N, D)$ is the shortest path from N to D that may be found when done from S's perspective (so that the found path does not loop back through S).

To determine which neighbors may be used as loop-free alternates, it is necessary to know $\text{Distance}_{opt}(N_i, S)$ for each S's neighbor N_i, $\text{Distance}_{opt}(S, D)$, and $\text{Distance}_{S}(N, D)$.

$\text{Distance}_{opt}(N_i, S)$ may be determined by running the Dijkstra algorithm withS as the source using reverse link costs; then the shortest paths reported by each node to S will be the total distance from the node to the source S instead of from the source S to the node. Alternatively, it may be determined by running the Dijkstra algorithm from each of the N_i.

$\text{Distance}_{opt}(S, D)$ may be determined by a Dijkstra algorithm with the normal link costs; it may also be determined as a side-effect of the Extended Dijkstra algorithm described below.

$\text{Distance}_{S}(N_i, D)$ may determined by the Extended Dijkstra algorithm . It is not sufficient to record all the sub-optimal paths found during the Dijkstra algorithm; due to the order in which nodes are probed, sub-optimal paths may not be correctly propagated to all relevant nodes in the network. Therefore, it is necessary to obtain the optimal path via each neighbor N_i to each node in the network. To do this, the Dijkstra algorithm is extended so that paths are explored via each neighbor. Unlike the original Dijkstra algorithm, which compares all paths and retains only the shortest, the Extended Dijkstra algorithm stores paths through each neighbor N_i in each node and compares two paths only if their first hop is the same neighbor. Additionally, to ensure that the paths are properly propagated to all nodes in the network, it is necessary to ensure that the shortest path via each first-hop neighbor N_i is explicitly passed from a node to its neighbors when it may no longer be shortened.

The following well-known algorithm for determining the least-cost path between two given nodes is known as Dijkstra's algorithm.

To determine the least-cost path between the node 201, as shown in Fig. 2, and the rest of the nodes 202-209, one divides the entire set of nodes into two non-overlapping sets 230 and 231. At the start of the process, only the starting node 201 belongs to the set 230, while all the other nodes belong to the set 231. To determine the next node to add to the set 230, choose the cheapest link leading from any member of the set 230 to any member of the set 231. In Fig. 2, the cheapest link is selected between the links 216-219. The path thus established to a node in the set 231 is selected as the least-cost path to that node, and the node is removed from the set 231 and placed into the set 230. In this manner, all nodes are moved one by one to the set 230 and the least-cost paths are established to each node. The proof why this process does indeed produce the least-cost paths to each node is well known among those skilled in the pertinent art and is outside the scope of this invention.

One way of using the Dijkstra algorithm to determining loop-free nodes, in particular for determining $\text{Distance}_S(N, D)$ for all N_i, is illustrated in Fig. 5. For each node 502-509 the algorithm produces a least-cost path that starts at 502, 503, or 504, and then goes to another node without ever going through 501. One way of doing this is to run a Dijkstra algorithm three times with links 512, 513, and 514 removed: first starting with 502, then starting with 503, and finally starting with 504. This produces $\text{Distance}_S(N_i, D)$ for each N_i: 502, 503, and 504, with D being a node 505-509.

The above method determines $\text{Distance}_s(N_i, D)$ for each neighbor N_i and each destination node D. This allows determination of whether a neighbor is loop-free with respect to S and a destination D.

The Dijkstra algorithm may be used to perform all the necessary calculations for embodiments of this invention. However, an Extended Dijkstra method may also be used as described in U.S. Provisional Application "Network Routing Algorithm" by

Alia Karin Atlas and Raveendra Torvi, filed on February 6, 2004, incorporated herein by reference.

The following provides additional information on functioning of embodiments of this invention.

1. Introduction

Applications such as VoIP and pseudo-wires can be very sensitive to traffic loss, such as occurs when a link or router in the network fails. A router's convergence time is generally on the order of seconds; the application traffic is sensitive to losses greater than 10s of milliseconds. This document describes a mechanism to allow a router whose local link has failed to forward traffic to a pre-computed alternate until the router installs the new primary next-hops based upon the changed network topology.

With current IP routing and forwarding, there is a non-trivial period of traffic loss in the event of a link failure. Consider the example topology shown below in Figure 6. In this example (and all other examples given in here), the green arrows show the shortest path tree towards the destination (D in this topology).

If the link from node S to node P fails, then traffic from S which is destined to D will be dropped until S recomputes and installs new forwarding state. This process of recomputing the shortest path (running an SPF algorithm) and installing the results can take seconds. This means that the traffic convergence can take seconds.

The goal of RAPID is to reduce that traffic convergence time to tens of milliseconds by having pre-computed an alternate interface to use, in the event that the currently selected primary interface fails, and having installed that alternate interface into the forwarding plane so that it can be rapidly used when the failure is detected.

This document describes RAPID from the perspective of the externally visible changes to a router, its associated protocols, and the interactions this causes in the network. It does not address any algorithms to compute the alternate next-hops nor any router internals to actually implement RAPID.

To clarify the behavior of RAPID a bit more, consider the topology shown in Figure 6. When Router S computes its shortest path to Router D, Router S determines to use the interface to router P as its primary next-hop. Without RAPID, that is all the only next-hop that router S computes to reach D. With RAPID, S also looks for an alternate next-hop to use. In this example, S would determine that it could send traffic destined to D by using the interface to router N_1 and therefore S would install the interface to N_1 as its alternate next-hop. At some point later, the link between router S and router P could fail. If that link fails, S and P will be the first to detect it. On detecting the failure, S will stop sending traffic destined to D towards P and the failed link and instead send the traffic to S's pre-computed alternate next-hop, which is the interface to N_1. As with the primary next-hop, the alternate next-hop is computed for each destination.

The terminology needed to understand RAPID will be described in Section 2. The different types of alternates and how to select an alternate are discussed in Section 3. The use of an alternate for breaking U-Turns and for protecting against a local failure is covered in Section 4 The external signaling required to support RAPID and the associated protocol extensions are discussed in Section 5 Section 6 discusses how the alternates which are computed for a single IGP area or level are inherited to the different protocols and types of routes. The interactions with IGP tunnels, RFC 3137, ISIS overloaded routers, and LDP/IGP Interactions are also described.

2. Terminology

The following describes the terminology used in describing alternates and finding alternates. The terms are introduced as they are used, but they are gathered together here for clarity.

SPT – Shortest Path Tree

SRLG – Shared Risk Link Group. This is a set of links which are dependent upon a common resource and may therefore fail at the same time. For example, multiple links may use the same router module and thus be part of a shared risk group.

localSRLG_M – Any SRLG which contains only links from router M.

D – The destination router under discussion.

S – The source router under discussion. It is the viewpoint from which RAPID is described.

P – The router which is the primary next-hop neighbor to get from S to D. Where there is an ECMP set for the shortest path from S to D, these will be referred to as P_1, P_2, etc.

N_i – The ith neighbor of S.

$R_{i,j}$ – The jth neighbor of N_i, the ith neighbor of S.

$\text{Distance}_{!S}(N_i, D)$ – The distance of the shortest path from N_i to D which does not go through router S.

$\text{Distance}_{opt}(A, B)$ – The distance of the shortest path from A to B.

Reverse Distance of a node X – This is the $\text{Distance}_{opt}(X, S)$.

Loop-Free Alternate – This is a next-hop that is not a primary next-hop whose shortest path to the destination from the alternate neighbor does not go back through the router S which may use it as an alternate.

U-Turn Alternate – This is an alternate next-hop of S that goes to a neighbor N_i, whose primary next-hop is S, and whose alternate is loop-free with respect to S and N_i.

Link(A->B) – A link connecting router A to router B.

Primary Neighbor - One or more of the primary next-hops for S to reach the destination D goes directly to this neighbor.

Loop-Free Neighbor – A Neighbor N_i which is not the primary neighbor and whose shortest path to D does not go through S.

U-Turn Neighbor – A neighbor N_i is a U-Turn neighbor of router S with respect to a given destination D if and only if S is a primary next-hop of N_i to reach the destination D for all primary paths which go through S to reach D.

ECMP U-Turn Neighbor - A neighbor N_i which is a U-Turn neighbor and which has at least one equal cost path to reach D that does not go through S as well as the path(s) which do go through S to reach D.

Loop-Free Node-Protecting Alternate – This is a path via a Loop-Free Neighbor N_i which does not go through any of S's primary neighbors to reach the destination D.

Loop-Free Link-Protecting Alternate – This is a path via a Loop-Free Neigbhor N_i which does go through one or more of S's primary neighbors to reach the destination D.

U-Turn Node-Protecting Alternate – This is a path via a U-Turn Neighbor N_i which does not go through S or any of S's primary neighbors to reach the destination D.

U-Turn Link-Protecting Alternate – This is a path via a U-Turn Neighbor N_i which does not go through S but does go through one or more of S's primary neighbors to reach the destination D.

Upstream Forwarding Loop – This is a forwarding loop which involves a set of routers, none of which are directly connected to the link which has caused the topology change that triggered a new SPF in any of the routers.

Solid arrow is indicating the primary next-hop towards the destination D.

Long-dash arrow is indicating the new primary next-hop towards the destination D in the event that the link between router S and router P has failed.

Short-dash arrow is indicating the alternate next-hop towards the destination D

3. Finding an Alternate to Use

3.1 Failure Scenarios

For very fast fail-over times using all known schemes, iIt is necessary to find an acceptable alternate to use before the failure occurs. What constitutes an acceptable alternate depends on what types of failures are to be protected against.

The simplest case is to protect against a single link failure.

However, a link, as seen in the IP topology, may not be independent of other links as seen in the IP topology. This may be because of multiple logical interfaces, such as VLANs on a Gigabit Ethernet interface or PVCs on an ATM port. It may also be because of channelization, so that multiple interfaces use the same physical fiber. It may also be because multiple links use the same internal hardware, such as a line-card, and so have a correlated failure. Each of these scenarios represent links local to the router which need to be grouped into correlated failure groups. Such a local failure group will be referred to as a local SRLG (Shared Risk Link Group). A local SRLG which contains links local to a router M will be indicated as localSRLG$_M$.

To clarify about protection of a local SRLG, consider the topology shown in Figure 12. In this example, S has a local SRLG which contains Link(S->P) and Link(S->N$_1$). Similarly, P has a local SRLG which contains Link(P->S) and Link(P->N$_2$). It is possible for S's alternate to only protect against S's own local SRLG and not against P's local SRLG; in that case, S might select N$_2$ as an alternate. If S's alternate also tried to protect against P's local SRLG, because the link from S to P was included in it, then S could not select N$_2$ or N$_1$ and so would need to select N$_3$, because N$_3$, to reach D, does not go via a link in the same local SRLG of P of which the potential failed link from S to P is a member. The former case will be referred to as providing local SRLG protection. The latter case will be referred to as also providing protection for the primary neighbor's local SRLG.

It is also useful to protect against a node failure.

One can generalize the concern about correlated failures to apply to links which are not local to a single router but instead contain any links from the topology. This allows considerations about physical equipment, such as conduits, fibers, etc., to be accurately expressed as correlated failures. These correlated failure groups which contain an arbitrary set of links are referred to as SRLGs (Shared Risk Link Groups). RAPID can provide link protection, node protection or local SRLG protection, depending upon the selection of the alternates which is done when the RAPID algorithm is run at a router S. General SRLGs are not considered for protection.

3.2 Types of Alternates

As with primary next-hops, an alternate next-hop is discussed in relation to a particular destination router D. For this discussion, the following terminology, illustrated in Fig. 8, will be used. The router on which the search for an alternate is proceeding is S. The primary next-hop neighbor to get from S to D is P. Additionally, S has various neighbors which will be labeled N$_1$, N$_2$, etc. Where an arbitrary neighbor of S is intended, N$_i$ will be used. Routers which are neighbors of neighbors will be labeled R_1, R_2, etc. Where an arbitrary neighbor of a neighbor N_i is intended, it will be referred to as $R_{i,j}$.

In standard IP routing, a router S can join the shortest path tree (SPT) at exactly one point – itself. An alternate next-hop allows traffic from S to D to deviate from the SPT and then rejoin it. For instance, if S were to send traffic destined for D to N_1 instead of P, thereby deviating from the SPT, then when N_1 received it, N_1 would send that traffic along its shortest path to D.

3.2.1 Loop-Free Alternates

To expand the set of points at which S can cause its traffic to join the SPT, first consider S's neighbors. Router S has the ability to send traffic to any one of its neighbors N_i; this is the easiest possible deviation from the SPT that S can cause to happen. Thus, all of router S's neighbors are possible points at which S could cause traffic to rejoin the SPT. However, it is not useful for router S to use an next-hop which results in rejoining the SPT upstream of S, such that the traffic will transit S again. This would cause a loop. Avoiding a loop is thus the first constraint imposed on the alternate next-hop. In Fig. 8, this is the case for S's neighbors N_2 and N_3.

A next-hop which goes to a neighbor that does not have a loop back to S and is not the primary next-hop may be selected as an alternate next-hop. In Fig. 8, that is the case for S's neighbor N_1. Such alternates are referred to as loop-free alternates because there is no loop caused by using them.

An algorithm run on router S must be able to determine which neighbors provide loop-free alternates. By running an SPF computation from S's perspective, router S can determine the distance from a neighbor N_i to the destination D for the optimal path that does not go through S. This is referred to as $\text{Distance}_{!S}(N_i, D)$. If a neighbor N_i can provide a loop-free alternate, then it is cheaper to get to the destination without going through S than by going through S. This gives the following requirement, where $\text{Distance}_{opt}(A, B)$ gives the distance of the optimal path from A to B.

$$\text{Distance}_{!S}(N_i, D) < \text{Distance}_{opt}(N_i, S) + \text{Distance}_{opt}(S, D)$$

Equation 1: Criteria for a Loop-Free Alternate

Recall that a router will take the shortest path to a destination that it can see. Thus, if $\text{Distance}_{!S}(N_i, D) > \text{Distance}_{opt}(N_i, S) + \text{Distance}_{opt}(S, D)$, then router N_i will, based on its own shortest path computations, determine to send traffic destined for D to S. Similarly, if $\text{Distance}_{!S}(N_i, D) = \text{Distance}_{opt}(N_i, S) + \text{Distance}_{opt}(S, D)$, then router N_i has equal cost paths to the destination D where one or more of those paths go through S. In such a case where a router N_i has an ECMP set to reach the destination and one or more paths go through S, then the router N_i cannot provide a loop-free alternate because some traffic destined to D may be sent back to S by N_i. Thus, if N_i is to decide not to send traffic for D back to S, N_i must know that the shortest path to D does not go through S; Equation 1 gives this requirement in terms which can be determined by router S.

3.2.2 U-Turn Alternates

In examining realistic networks, it was seen that loop-free alternates did not provide adequate coverage for the traffic between all the source-destination pairs. This means that it is not sufficient to expand the set of points where S can cause its traffic to join the SPT to be S's neighbors.

The next possibility is to see whether S could expand its SPT join points to include router S's neighbors' neighbors. This is only of interest if S had no loop-free node-protecting alternate available for the given destination D. If there are no loop-free alternates, that implies that all of S's non-primary neighbors will send traffic for D back to S.

The topology shown in Fig. 9 gives an example where router S has no loop-free alternate to reach D. Router S uses N_1 as its primary next-hop (distance of 30). S has three other neighbors, but all of them will send traffic for D back through S.

In order for S to be able to use a neighbor's neighbor as a point where S's traffic can rejoin the SPT, S must be able to direct traffic to a neighbor N_i and that neighbor N_i must be able to direct traffic to one of its appropriate neighbors $R_{i,j}$ instead of along the SPT. In deciding to use its alternate, S has the ability to force traffic destined to D to go through the selected alternate neighbor N_i. However, for S to reach the appropriate neighbor's neighbor $R_{i,j}$, the selected neighbor N_i must be able to detect that the traffic should not be sent along its shortest path to D, which would lead back to S, and should instead be sent to its appropriate neighbor $R_{i,j}$.

This detection and forwarding contrary to the SPT by N_i must occur without any communication from S upon the failure which would cause S to redirect the traffic to N_i. There is already communication from S to N_i indicating when a link has failed, but such communication would cause the fail-over of traffic to take longer if N_i depended upon it to decide that it should forward contrary to the SPT. In essence, the assumption being made is that the time budget to recover traffic in the event of a failure is being consumed by router S's detection of the failure and switch-over to its pre-computed alternate.

With that assumption, it is clear that N_i's behavior to forward traffic contrary to the SPT on receiving traffic from S must be a default behavior. This default behavior must not change how traffic is forwarded unless a forwarding loop is detected; basic IP forwarding must be preserved in the absence of a failure. Router N_i can detect if it is receiving traffic from a neighbor to whom it would forward that traffic; this detection is done via a reverse forwarding check. Such a reverse forwarding check may only consider if traffic is received on the same interface as it would be forwarded out, but logically it should consider the neighbor and not merely the interface. Normally, if traffic fails a reverse forwarding check (i.e. would be forwarded out to the same neighbor as received from), then that traffic is normally either discarded or forwarded into a loop. In RAPID, however, traffic that fails a reverse forwarding check is forwarded to the appropriate $R_{i,j}$, if available, rather than being discarded.

First, this detection can be used by N_i to determine not to forward the traffic according to the SPF (or discard it), but to instead send the traffic to Ni's appropriate neighbor $R_{i,j}$. N_i can only detect the traffic to be redirected if S sends it directly to N_i, which is under S's control, and if N_i would send that traffic back to S, according to the SPT. This motivates the definition of a looping neighbor and a U-turn neighbor.

Looping Neighbor - A neighbor N_i is a looping neighbor of router S with respect to a given destination D if any of N_i's shortest paths to D goes through S but S is not the primary next-hop of N_i for all those paths through S.

U-Turn Neighbor - A neighbor N_i is a U-Turn neighbor of router S with respect to a given destination D if and only if S is a primary next-hop of N_i to reach the destination D for all primary paths which go through S to reach D.

For a Looping Neighbor to provide an alternate would require changing the forwarding state associated with links from any neighbor which an optimal path to D traversed; additionally, appropriate alternates which avoided that neighbor would be necessary to compute. This would cause the complexity of RAPID to increase. Therefore for this version, we disallow using an alternate via a Looping Neighbor. A U-Turn neighbor may be able to provide an alternate. In Figure 4, S has two U-Turn Neighbors N_2 and N_3 and one looping neighbor N_4. For neighbor N_4, the path to D is N_3 to S to N_1 to R_1 to D; because there is a node between N_4 and S on the path, N_4 is a looping neighbor.

Mathematically, for a neighbor Ni to be a U-Turn neighbor, it is necessary that Equation 2, which is the exact opposite of Equation 1, be true. If the equality is true, that means that there are multiple optimal paths, at least one of which goes through S and one does not. Such a neighbor may be an ECMP U-Turn neighbor or may be a looping neighbor.

$$\text{Distance}_{!S}(N_i, D) = \text{Distance}_{opt}(N_i, S) + \text{Distance}_{opt}(S, D)$$

Equation 2: U-Turn or Looping Neighbor

Additionally, all optimal paths to reach D that go via S must be via a direct link between N_i and S. If a neighbor N_i satisfies Equation 2 and all optimal paths to reach D that go via S are via a direct link between N_i and S, then it is a U-turn neighbor.

The above clarifies what a U-Turn neighbor is and how such a neighbor can detect traffic from router S and redirect it. It is still necessary to describe where the U-Turn neighbor N_i redirects the traffic.

3.2.2.1 ECMP U-Turn Neighbors

The above definition for U-Turn Neighbor allows a neighbor, which has equal cost paths (an ECMP set) where one of those paths goes directly to S and others may not, to be a U-Turn Neighbor. Consider the topology shown in Figure 10. In this figure, N_1 has three equal-cost paths to reach D which are N_1 - S - P - D, $N_1 - R_1 - D$, and $N_1 - R_2 - D$. Because the only path that goes through S goes directly through S, N_1 is a U-Turn neighbor.

$$\text{Distance}_{!S}(N_i, D) = \text{Distance}_{opt}(N_i, S) + \text{Distance}_{opt}(S, D)$$

Equation 3: ECMP Neighbor

A neighbor is an ECMP neighbor if Equation 3 is true. The complication comes because S does not whether a neighbor N_i supports ECMP or how that neighbor selects among the equal cost paths. Recall that a node will only break U-Turns on the interfaces connected to that node's primary neighbors.

Consider the topology in Figure 11, where N_2 has three equal cost primary neighbors which are S, N_1 and A. If N_2 were to select only N1 as its primary neighbor, then N2 would break U-Turns only on traffic received from N1 and not on traffic received from S. Therefore, S cannot consider N2 as an ECMP U-Turn neighbor because S cannot rely upon N2 to break U-turns for traffic destined to D which is received from S.

If N2 has multiple paths to reach D which go through S and not all such paths have a first hop which is a direct link between N2 and S, then S cannot use N2 as a U-Turn neighbor because N2 in this case is a Looping Neighbor.

If all paths from an ECMP neighbor Ni to destination D which go via S have S as the primary neighbor, then S can use N2 as a ECMP U-Turn neighbor.

3.2.2.2 U-Turn Neighbor's Alternate

The requirement for the neighbor's neighbor Ri,j to which a U-Turn Neighbor Ni will redirect traffic from S destined to D is that the traffic will not come back to S. Equation 4 gives this requirement that Ri,j must have a path to D that does not go through S which is shorter than the path to D going via S. This can be expressed as follows.

Distance!S(Ri,j, D) < Distanceopt(Ri,j, S) + Distanceopt(S, D)

Equation 4: Loop-Free Neighbor's Neighbor

Equation 4 means that a U-Turn neighbor's alternate cannot be an ECMP set which contains that U-Turn neighbor.

If Ni is a U-Turn neighbor, then the optimal path to D from Ni is via S; the path is Ni - S - ... - D. Therefore, if the optimal path from Ri,j goes through Ni, it must also go through S. Thus, if Equation 4 holds for a Ri,j, that implies that the path from Ri,j does not go through Ni. This may be made clearer by considering Figure 12 below. If the shortest path from R to D went through N1, then it would go through S as well, because the shortest path from N1 to D is through S. Therefore, if the shortest path from R does not go through S, it cannot have gone through N1.

Proof 5: Proof that a Loop-Free Ri,j (Neighbor's Neighbor) Implies Ri,j Doesn't Loop to Neighbor Ni

The proof given in Proof 5 means that if a U-Turn Neighbor Ni has itself a neighbor Ri,j that satisfies Equation 4, then that router Ri,j is itself a loop-free alternate with respect to Ni. Regrettably, the converse does not apply; just because Ri,j is loop-free with respect to Ni and D does not mean that Ri,j is loop-free with respect to S and D.

3.2.2.2.1 Computing Alternate such that the Primary Next-Hop Can Use the Computing Router as U-Turn Alternate

Each router independently computes the alternate that it will select. It is necessary to consider what alternate S could select so that S's primary next-hop P could use S as a U-Turn alternate. In other words, consider the computation when S is in the role of a neighbor to the router doing the computation.

To describe this using router S as the computing router, S would need to verify that both Equation 1 is true and that S's selected alternate Ni does not have a path that goes through P.

This can be described as if Ni were doing the computation as follows. The criteria described in Equation 4 requires that if a U-Turn neighbor Ni is to be used as a

U-Turn alternate then Ni must have a loop-free alternate which avoids Ni's primary neighbor S. Such an alternate will be referred to as a loop-free node-protecting alternate. Ni can identify loop-free alternates by checking the validity of Equation 6. Additionally, Ni will need to tell whether the path from a loop-free Ri,j to D goes through Ni's primary next-hop neighbor, S.

Distance!S(Ri,j, D) < Distanceopt(Ri,j, Ni) + Distanceopt(Ni,, D)

Equation 6: Neighbor's Loop-Free Alternate

3.2.3 Proof that U-turn Alternates Do Not Cause Loops

One can exhaustively go through all of the possibilities. Assume a topology such as shown in Fig. 13. RAPID protects against the failure of Link(S P) by providing an alternate path through N. Once Link(S P) fails, S forwards traffic destined for D to N.

Either N or R are on a shortest path to D that does not go via S. If N is a loop-free alternate of S, it is on a shortest path to D that does not include S. If N is a U-Turn alternate of S, then R is on a shortest path to D that does not include S.

N is either a loop-free neighbor of S or it is a U-Turn neighbor of S. If N is a loop-free neighbor of S, then all traffic destined for D through N will not travel through S. Thus there is no loop caused by the redirection of traffic by S to N because N will not forward the traffic back to S.

If N is a U-Turn neighbor of S, traffic destined for D from S to N will be forwarded to R, but traffic destined for D and not from S will be forwarded to S. Thus, there are two cases: traffic destined for D either (i) goes through S before N OR (ii) goes through N before S.

In case (i), there is no loop because N will forward the traffic to R.

In case (ii), there is what appears to be a loop (N->S->N). Due to its separate forwarding table for traffic arriving from S, however, N has effectively become two nodes with the first node (N) being exactly the same as the original N and the second node (N') having two links, one to/from S and the other to/from R. Since N becomes N' for traffic from S to D, the path now becomes N->S->N'. N' forwards traffic to D through R which is on a shortest path to D that does not include S.

To prove freedom from loops, one can simply show that any given node is only visited once. However, U-turn alternates are visited more than once. Since those nodes behave differently based on whether the traffic enters via their primary or not, they are effectively two nodes for the given destination D, the first node (X) being exactly the same as the original node with a single routing table for the destination while the second node (X') has two links, one to its primary and the other to its alternate. Each of the two nodes will be traversed at most once, since the first node X will forward towards its primary and the second node X' will forward towards the alternate. The alternate path is not taken until S is hit and the U-turn commenced. There is no crossover that would create a loop since all of the first nodes are hit before any second node, since the second nodes are only on the alternate path and cannot be reached from the primary path except through S. Thus, the graph is acyclic and will have no loops.

3.3 Selection of an Alternate

A router S may have multiple alternates that it must decide between. A common selection method is necessary to support U-Turn Alternates. This is because it is not sufficient for router S to know that its U-Turn neighbor Ni has itself a neighbor Ri,j that is loop-free with respect to S and D if S does not also know that Ni will select that Ri,j or another with the same properties.

3.3.1 Configuration Control: RAPID Alternate Capability

There are a number of different reasons why an operator may not wish for a particular interface to be used as an alternate. For instance, the interface may go to an edge router or the interface may not have sufficient bandwidth to contain the traffic which would be put on it in the event of failure.

If an interface cannot be used for an alternate, then the interface will be have its RAPID Alternate Capability will be false and otherwise it will be true.

3.3.2 Interactions with Maximum Costed Links

A router may advertise itself as overloaded, for ISIS, or indicate a link weight of LSInfinity (for OSPF), or the equivalent maximum weight (for ISIS). This is done in several circumstances.

First, the operator is intending a maintenance window for the interface or router and the operator does not want any transit traffic to be directed across that link or through that router. The link or router is kept active in the topology so that the link's or router's local addresses can still be reached.

Second, the router has not learned the necessary information to be able to accurately forward a subset of traffic, either BGP-distributed prefixes or LDP FECs. For the case of BGP, the entire router will be indicated to not be used for transit. In the case of LDP, one or more links will be set to the maximum cost to avoid that link being used to transit LDP traffic.

RAPID must respect the intentions of having a link set to maximum cost and/or a router being overloaded. This is particularly required because those links would otherwise look very tempting to RAPID, because the Dopt(Ni, S) would be quite large if Ni has set the links between itself and S to the maximum cost.

Therefore, when looking for alternates, a router S cannot consider diverting from the SPT to a neighbor Ni if all links between S and Ni have a maximum reverse cost or if Ni is overloaded. Similarly, router S cannot consider that a neighbor Ni could provide a U-turn alternate via a neighbor's neighbor Ri,j when Ri,j is overloaded or if all the links between Ni and Ri,j have a maximum reverse cost.

3.3.3 Characterization of Neighbors

Each neighbor Ni must be categorized as to the type of path it can provide to a particular destination. Each neighbor can be characterized as providing a path in one of the following categories for a particular destination D. The path through the neighbor Ni is either a:

(A) Primary Path - one of the shortest paths that is selected as a primary next-hop,

(B) Loop-Free Node-Protecting Alternate - not a primary path and the path avoids both S, the interfaces connecting S to its primary neighbors, and its primary neighbors on the path to D.

(C) Loop-Free Link-Protecting Alternate - not a primary path and the path avoids S and the interfaces connecting S to its primary neighbors, but goes through a primary neighbor on the path to D.

(D) U-Turn Node-Protecting Alternate - the neighbor is a U-Turn neighbor or a ECMP U-Turn neighbor and the alternate that the neighbor has selected does not go through a primary neighbor of S to reach D.

(E) U-Turn Link-Protecting Alternate - the neighbor is a U-Turn neighbor or a ECMP U-Turn neighbor and the alternate that the neighbor has selected goes through a primary neighbor of S to reach D.

(F) Unavailable - because the neighbor is looping or a U-Turn neighbor which didn't itself have a loop-free node-protecting path, or a U-Turn neighbor which couldn't break U-Turns or the links to the neighbor are configured to not be used as alternates. The neighbor may also be disqualified because the link to reach it is in a local SRLG with the primary next-hop. The neighbor may be connected to S via a broadcast interface which is a primary next-hop.

3.3.4 Selection Procedure

Once the neighbors have been categorized, a selection can be made. The selection should maximize the failures which can be protected against. A node S can only be used to break U-turns by its primary neighbors if S has a loop-free node-protecting alternate. This is a consequence of Equation 4 and assumes that multiple U-Turns cannot be broken in order to find an alternate; such extensions are for future study.

The selection procedure depends on whether S has a single primary neighbor or multiple primary neighbors. A node S is defined to have a single primary neighbor only if there are no equal cost paths that go through any other neighbor; i.e., a node S cannot be considered to have a single primary neighbor just because S does not support ECMP.

3.3.4.1 Alternate Selection With a Single Primary Neighbor

Because a node S can only be used to break U-Turns by its primary neighbor if S selects a loop-free node-protecting alternate, the following rules should be followed when selecting an alternate. This describes a policy which allows U-Turn alternates to function in a way which is more efficient given our current algorithm; other policies are possible and will allow U-Turn alternates to function.

1. If a node S has one or more loop-free node protecting alternates, then S should select one of those alternates. Let M be the set of neighbors which provide loop-free node-protecting alternates. If S has multiple loop-free node protecting alternates, then S should select the alternate through a Nk such that:

$$\mathrm{Dopt}(Nk, D) - \mathrm{Dopt}(Nk, P) = \min\ m\ M\ (\mathrm{Dopt}(m, D) - \mathrm{Dopt}(m, P))$$

Equation 7: Selection Among Multiple Loop-Free Node-Protecting Alternates

where P is the primary neighbor of S.

To rephrase the above consider S is the node looking for a U-Turn alternate. Using Equation 7 to select among loop-free node-protecting alternates ensures that N_i's primary neighbor S can determine which alternate was picked by N_i. For S to know that S's U-Turn neighbor N_i can provide a loop-free node-protecting alternate, S must know if

$$\min\ \ j \text{ in } J\ \ (D!S(Ri,j\ , D) - \mathrm{Dopt}(Ri,j, S)) < \mathrm{Dopt}(S, D)$$

Equation 8: Determination if a U-Turn Neighbor can provide a U-Turn Alternate

If a router obeys Equation 7 when selecting among multiple loop-free node-protecting alternates, as it MUST for IP/LDP Local Protection, this allows S to determine exactly which alternate was selected by N_i without needing to know the each D_!S(R_i,j). Equation 8 allows S to determine that N_i has a loop-free node-protecting alternate. Equation 7 allows S to know exactly which alternate will be selected so that S can determine whether that alternate protects against S's primary neighbor as well. If there are multiple neighbors which provide the minimum as expressed in Equation 7, then a router can select among them arbitrarily.

To rephrase the above to consider the S is the node looking for a U-Turn alternate, the above way of selecting among loop-free node-protecting alternates ensures that Ni's primary neighbor S can determine which alternate was picked by Ni. For S to know that S's U-Turn neighbor Ni can provide a loop-free node-protecting alternate, S must know if min j J (D!S(Ni,j, D) - Dopt(Ni,j, S)) < Dopt(S, D).

2. If a router S has no loop-free node-protecting alternates, then S's alternate selection has no consequences for its neighbors because S cannot provide a U-Turn alternate. Therefore, S can select freely among the loop-free link-protecting alternates, u-turn node-protecting alternates and u-turn link protecting alternates which S has available. Clearly selecting a u-turn node-protecting alternate, if one is available, will provide node-protection, while the other options will not. Selection among these categories is a router-local decision.

3. If S has neither loop-free node-protecting alternates, loop-free link-protecting alternates, u-turn node-protecting alternates, nor u-turn link-protecting alternates, then S has no alternate available for traffic to the destination D from the source S.

3.3.4.2 Alternate Selection With Multiple Equal Cost Neighbors

The selection among multiple equal cost paths is a router-local decision. Therefore, a router N_i cannot know which of the potential primary neighbors that S will choose to use.

As described in Section 3.2.2.2 , N_i can only select S for its U-Turn alternate if any potential primary neighbor which S might select, except for N_i itself, will not go via N_i to reach the destination D.

Since a router S has multiple potential primary neighbors, router S MUST select one or more alternates for breaking U-Turns from among next-hops to its potential primary neighbors. If router S does not have a potential primary neighbor that is node-protecting for a particular primary next-hop, that indicates that the particular primary neighbor will not use S as a U-turn alternate.

Router S need not use the same alternate(s) for breaking U-Turns on traffic received from a primary next-hop as for when the primary next-hop fails. The alternate(s) used when a primary next-hop fails are a router-local decision.

4 Using an Alternate

If an alternate is available, it is used in two circumstances. In the first circumstance, it is used to redirect traffic received from a primary next-hop neighbor. In the second circumstance, it is used to redirect traffic when the primary next-hop has failed. As mentioned in Section 3.3.4.2 , for destinations with multiple potential primary neighbors, the alternates used for each purpose need not be the same.

4.1 Breaking U-Turns

If one ignores potential security redirection, IP forwarding is a purely destination based algorithm. Traffic is forwarded based upon the destination IP address, regardless of the incoming interface.

As previously described in Section 3.2.2 , RAPID requires that a U-Turn neighbor be capable of detecting traffic coming from the primary next-hop neighbor and redirecting it to the alternate, if an alternate which is node-protecting is available. This becomes the new default behavior. This behavior is described in Fig. 14.

Table 1 : Forwarding Rules for Traffic From Primary Next-Hop

The rules described in Fig. 15 apply to traffic received on an interface whose primary next-hop is the same interface. On point-to-point interfaces, it never makes sense to send the traffic back to the primary next-hop because that will go to the primary neighbor and always cause a forwarding loop. Therefore, unless an interface is U-Turn capable and has a loop-free node-protecting alternate, traffic received on its primary next-hop will be discarded. If an interface is U-Turn capable and has a node-protecting alternate, traffic received on its primary next-hop will be forwarded to its alternate next-hop.

Any link, including a broadcast link, must belong to one area (in OSPF) or level (in ISIS); one cannot have subset of nodes for a given network segment in different areas. If a broadcast interface is U-turn capable, then it is acceptable to forward traffic from all nodes on that interface via the alternate path.

Consider the topology example shown in Fig. 10. In this case, router N1 has a primary and an alternate for two destinations D and C. The primary next-hop for destination D is router S and the alternate next-hop is R1. Similarly, the primary next-hop for destination C is router R1 and the alternate next-hop is R2. The three interfaces L1, L2, and L3 shown on router N1 have different forwarding tables as shown in Fig. 16; additional interfaces would have the same forwarding table as for interface L2, which is not a primary next-hop for either destination.

The ability to break U-Turns may vary depending upon the type of traffic - IP or MPLS. By default, broadcast interfaces will not be administratively configured as U-Turn capable until explicitly configured. Point-to-point interfaces will be administratively configured as U-Turn capable by default. An interface is U-Turn capable for a particular type of traffic if the interface is administratively configured as U-Turn capable and if the interface hardware can break U-Turns for that type of traffic.

4.1.1 Broadcast and NBMA Interfaces

NBMA and broadcast interfaces can be treated identically for RAPID; both involve the case of possibly receiving traffic from multiple neighbors. With broadcast interfaces (i.e. Gigabit Ethernet), there can be multiple neighbors connected to the same interface. It is extremely desirable to have at most one forwarding table per interface. Therefore, it must be considered whether all traffic received on an interface can be treated identically, regardless of the neighbor sourcing the traffic on that interface.

The cost for any node on the broadcast interface to reach S or P will be identical. Because all link costs are positive, no neighbor on the broadcast interface will ever send traffic to S along that interface in order to reach P. Therefore, S can assume that any traffic received on the broadcast interface which goes to a destination via a primary next-hop neighbor that is also on the broadcast interface is in fact sent by that primary next-hop neighbor and should be redirected to break the U-Turn.

Thus, if router S has a primary next-hop neighbor for a given prefix on the broadcast interface, S should redirect all traffic received destined to that prefix on the broadcast interface to S's alternate next-hop.

An interface can be either a primary next-hop or the alternate next-hop, but not both because there would be no protection if the interface failed.

4.2 Responding to a Failure

When a failure is detected, traffic which was destined to go out that failed interface must be redirected to the appropriate alternate next-hops. The alternate next-hop is calculated to be reliable in the event of the failure scenario being protected against.

RAPID does not attempt to add anything new to the detection of the failure. The same mechanisms that work for RSVP-TE Fast-Reroute can work here. For SONET interfaces, this means detecting a failed link immediately, rather than waiting the standard 2 seconds. For Gigabit Ethernet, for directly connected interfaces, RFI can be used; other mechanisms can be investigated as appropriate and necessary.

Because the alternate next-hop is pre-computed and can be available on the forwarding plane, it should be extremely fast to switch traffic to use it. This can be the same mechanisms used for Fast-Reroute.

4.3 Avoiding Micro-Forwarding Loops

4.3.1 For Loop-Free Alternates

There are three different scenarios for where router S will send traffic after its primary next-hop link fails. In the topologies shown, the green arrows show the initial SPT towards destination router D, the red arrow shows the alternate computed for the initial SPT, and the blue arrows shows the new SPT after the link between router S and router P has failed.

In the first scenario, router S will direct traffic to a neighbor which was loop-free with respect to S both before and after the link fails. Consider the topologies shown in Fig. 18. In the topology on the left, router S determines that it should send traffic destined to D to N1 after recomputing its shortest paths; in the topology on the right, router S determines to send the traffic destined to D to N2 after recomputing. In the left topology, it doesn't matter when S updates its forwarding table from using the alternate to using the new primary because they are the same. In the right topology, the alternate and the new primary are different, but both the path from the alternate neighbor and the path from N2 (the new primary neighbor) are loop-free with respect to S. Because the path from N2 was not dependent on the failed link, whether N2 or other routers on its shortest path to D have recomputed based on the changed topology doesn't matter because the results of the computation are the same for the previous and the new topologies.

In the second scenario, router S will direct traffic to a neighbor which was a U-Turn neighbor. Here there are two sub-scenarios. The new primary neighbor may not have been the alternate neighbor but has a loop-free alternate that also avoids S (left in Fig. 19). The new primary neighbor may not have been the alternate neighbor and has no alternate that also avoids S (right in Fig. 19).

In the topology on the right, router S's primary is P and its loop-free node-protecting alternate is N1. Router N2 has a loop-free alternate, N3, to protect against the link from N2 to S failing. When router S recomputes its shortest path to D after the link from S to P has failed, S decides to send the traffic to N2. If N2 has not recomputed and installed new forwarding, then N2 will discard the traffic because N2 had no alternate which could be used to break U-Turns; the router N3 provides a loop-free link-protecting alternate, not a loop-free node-protecting one. If N2 has recomputed, then the traffic will be sent to R1. Router R1 will forward traffic received from N2 destined to D directly to D before the topology change because R1 is breaking the U-Turn and afterwards because that is the new shortest path.

It is possible to construct additional scenarios like the one on the right in Fig. 19 where there is a dependency on a router further along the path having recomputed and installed new forwarding state. What is necessary is that, on failure of its primary next-hop, S switches to using a path which is independent of the failure and thus doesn't change as a result of the failure until the remainder of the network has converged. S's alternate is a path which is independent of the failure. Router S should keep using its alternate until the remainder of the network can be considered to have converged or until the failure scenario which is protected by RAPID has been violated. The failure scenario can be violated by S learning of a link failure elsewhere in the network that is not in a common local SRLG with the initial failed link. In our implementation, Router S assumes when the remainder of the network has converged based upon an operator-configured hold-down on new forwarding table installation. The only changes in the forwarding state that S will be waiting to install will be as a result of the local link failure and are thus exactly those which must be delayed.

The third scenario is when S's new primary is to a looping neighbor. This has sub-scenarios with the same issue as that for the scenario shown on the right in Fig. 19. The solution is the same, because the alternate is independent of the failure and thus doesn't change as a result of the failure.

4.3.2 For U-Turn Alternates

The hold-down discussed in Section 4.3.1 works because the alternate path being used does not change as a result of the failure. For a U-Turn alternate, the question is what path might the U-Turn neighbor converge to once it learns of the link failure. There are two possibilities for the new path from the U-Turn neighbor. Either the U-Turn neighbor Ni will converge to a path which uses S to reach the destination D or the U-Turn neighbor will converge to a path which does not use S to reach D. The former case is shown in the topology in Fig. 20 where router S is using a U-Turn alternate, N1.

Before the link between S and P failed, the shortest path for Ni to reach D went through S and then through P. If the link between S and P fails, this does not change the shortest path from Ni to S. Therefore, if after the failure, Ni's new path to reach D goes through S, it must go directly across the same link to S and therefore Ni will continue to consider traffic from S towards D to be in a U-Turn which requires breaking. Additionally, if Ni has a loop-free node-protecting alternate before the link between S and P failed, then Ni still has that loop-free node-protecting alternate, because it was not using the link from S to P since the alternate did not go through S. Thus, if Ni converges to a new path towards D which includes S, Ni will continue to use S as a primary neighbor and break U-Turns for the traffic received from S for D and Ni will continue to have a viable loop-free node-protecting alternate.

`If Ni converges to a path which does not include S to reach D, then traffic received from S for D will be sent along the new path and no forwarding loop will ensue.

4.3.3 Upstream Forwarding Loops

It is the nature of IP routing and forwarding that each router makes an independent computation and installs its forwarding state based upon its knowledge of the topology. This means that after a topology change, such as a link failure, each router may be forwarding based upon the old topology or the new topology until, eventually, all routers are forwarding based on the new topology. For ease of discussion, the following terms are introduced.

Unaffected Router - If a router has the same shortest path to the destination D before the link failure and after the link failure, then it is considered an unaffected router. It is still termed an unaffected router even if its alternate next-hop changes as a result of the failure.

Outdated Router - If a router will have a shortest path to the destination D after the link failure and the router has not yet installed the new shortest path into the forwarding plane, then it is considered an outdated router.

Updated Router - If a router will have a shortest path to the destination D after the link failure and the router has installed the new shortest path into the forwarding plane, then it is considered an updated router.

Affected Router - A router that will have a different shortest path to the destination D after the link failure is called an affected router. Such a router will be either an outdated or an updated router.

For a particular Affected Router X and a destination D, if the path from X to D encounters an Unaffected Router or S, then the traffic from X to D will not loop. If the above is true for all Affected Routers in the network, then there are no upstream forwarding loops.

With IP routing, there can be upstream forwarding loops, depending upon the convergence times of the individual routers in the network. It is possible for RAPID to assist in some of these upstream forwarding loops by breaking a loop between two neighbors, because that will be perceived as a U-Turn and RAPID will send the traffic to a loop-free node-protecting alternate, if available.

However, because RAPID will detect and attempt to break U-turns, it is possible that multiple single hop loops will be extended to form one longer loop. Consider the topology shown in Figure 21, where router A's new primary neighbor B is outdated and considers that router A is its primary neighbor. Router B will detect the U-turn when A updates its forwarding tables and, if router B has a node-protecting alternate, then B will direct the traffic to the node-protecting alternate, thereby breaking the loop. It is possible to construct a topology where multiple alternates will be used and a longer forwarding loop may therefore be created. In Figure 15, router B will break a U-Turn of traffic from A and send that traffic to C. C will send this traffic to E and E will detect this as a U-Turn which must be broken and therefore send the traffic along C's alternate back to A.

Extended upstream forwarding loops, such as in the above topology, can occur when two or more forwarding loops are joined together by two or more loop-free node-protecting alternates. For instance, in the example above, there are two forwarding loops; one is between A and B and the other is between E and C. For the first forwarding loop, B has a loop-free node-protecting alternate that it uses to try and break the loop; this sends the traffic to C. For the second forwarding loop, E has a loop-free node-protecting alternate that it uses to try and break the loop; this sends the traffic to A. Because of these two alternates, the two separate forwarding loops A-B and E-C are joined to create a larger forwarding loop A-B-C-E.

To avoid forwarding loops which could impact more links, it may be useful to analyze a topology and decide which interfaces should not break U-Turns and therefore can't provide a U-Turn alternate.

With RAPID, the timeline pictured in Fig. 22 describes the traffic loss in the event of a failure. As soon as the failure is detected, the alternate is switched into use. After a period of time (seconds), other routers in the network start installing their updated forwarding state. This starts the period where the upstream forwarding loops discussed in this section become a concern. Once the routers have converged, all traffic is forwarded properly. The period where the upstream forwarding loops are an issue depends on the difference in times for the routers to update their forwarding tables, exactly as without RAPID. What RAPID does do is preserve the traffic until the routers start to converge and then any traffic not caught in an upstream forwarding loop is preserved until the entire network has converged to the new topology, at which point the traffic can be forwarded normally.

For a given network, a given destination and a given link failure, it is interesting to consider which routers might forward traffic into loops when the new forwarding state begins to be installed.

4.4 Requirements on LDP Mode

In order for LDP to take advantage of the alternate next-hops determined, it is necessary for LDP to have the appropriate labels available for the alternate so that the appropriate out-segments can be installed in the hardware before the failure occurs.

This means that a Label Switched Router (LSR) running LDP must distribute its labels for the FECs it can provide to those neighbors which may require the labels, regardless of whether or not they are upstream. Additionally, LDP must be acting in liberal label retention mode so that the labels which correspond to interfaces that aren't currently the primary next-hop are stored. Similarly, LDP should be in downstream unsolicited mode, so that the labels for the FEC are distributed other than along the SPT.

5 Communicating with Neighbors

For loop-free alternates, there is no additional capability required on the part of the alternate next-hop neighbor. This is not the case for a U-Turn neighbor. In order to support U-Turn alternates, a router must know some details about its neighbors to know whether or not a U-Turn neighbor can act as a U-Turn alternate.

5.1 Providing RAPID Alternate Capability

One of the things that a router must determine about a U-Turn neighbor is whether that U-Turn neighbor has a loop-free node-protecting alternate. To know this, the router must know which interfaces the neighbor is able to use for alternates. This is administratively configured via the RAPID Alternate capability.

5.2 Communicating Capabilities for Breaking U-Turns

A router S cannot use a U-Turn neighbor as an alternate if that neighbor is not capable of detecting and breaking the U-Turn. One part of the ability to break the U-Turn is having an appropriate alternate, which router S can determine. The second part is that the neighbor support RAPID, so that it can have computed the appropriate alternate, and that the neighbor's forwarding plane is capable of detecting the U-Turn and breaking it.

If a router N advertises that it can break U-turns for a particular interface, then if N's primary neighbor S for a destination D is connected via that interface, then S can consider selecting that interface as a U-Turn alternate next-hop.

5.3 Distributing Local SRLGs

The following rules can be used to protect against local SRLG failures. Other methods are possible for protecting against arbitrary SRLG failures as well as local SRLG failures.

First, a link between S and Ni cannot belong to the same local SRLG as the link between S and P; if it did, then S could not consider U-Turn Neighbor Ni as an alternate. Second, no other relevant link on Ni can belong to the same local SRLG as the link between S and P; this is because no link to S will be used by the loop-free node-protecting alternate that Ni has, if it has any. Thus, only the local SRLGs of Ni need to be considered when deciding what interfaces are candidates for a loop-free node-protecting alternate.

There are already internet-drafts describing extensions to ISIS and OSPF to allow SRLGs to be signaled.

If a router considers local SRLGs when selecting an alternate, this affects which loop-free node-protecting alternates are available. Therefore, the router capability to consider local SRLGs during alternate selection should be signaled via a router capability TLV.

When local SRLG protection is supported, an additional bit to do so will need to be obtained and used in the router capability TLV.

5.4 Protocol Extensions for OSPF and ISIS

For an IGP, it is necessary to that a router know the neighbor's RAPID Alternate Capability for each of its interfaces. Additionally, a router must know whether the neighbor can break U-Turns for IP traffic on each of its interfaces which are directly connected to the router. This information can be propagated with a link-scope for flooding, as only the neighbors need to know this information.

5.4.1 OSPF and ISIS Extensions

One way to provide this information is as follows. The Router Capability TLV will have an additional bit defined for IP/LDP Local Protection. Additionally, two bits in a Link Capabilities sub-TLV will be defined; one bit will indicate that the interface can be used by the router as an alternate, while the other bit will indicate that the router can break U-turns on traffic coming into that interface. The same type of extensions can be used for both OSPF and IS-IS.

If a link is usable as an alternate, then the router's neighbors can assume that the router will have considered that link as an alternate next-hop.

See Figs. 23 and 24.

Then the router can determine if traffic received on that link is from the router's primary neighbor for that traffic and, if so, redirect it to the router's alternate next-hop. If a router's link is usable as a U-Turn recipient, then the router's neighbor can use select for an alternate a U-Turn alternate which goes across that link to that router.

5.5 Protocol Extensions for LDP

It may be desirable to signal the ability to break U-Turns separately for LDP traffic, in case there is hardware which can break U-Turns for MPLS traffic but not for IP or vice versa. If so, then the following extension could be used.

The only message which is sent and received on a per interface basis is the Hello message. It is possible to take one of the fourteen reserved and unused bits in the Hello

Message and use its being set to indicate that the interface is capable of breaking a U-Turn for MPLS traffic.

No LDP extensions will be necessary because all hardware which is capable of breaking U-Turns for IP traffic will be able to break them for MPLS traffic, and vice versa.

6 Routing Aspects

The RAPID algorithm is run for each topology, as represented by a link-state database. IGP protocols keep separate link-state databases for each process and for each area or level within a particular process. RAPID does not pass information across a process. The IGP protocols need to determine the inheritance of the RAPID alternates, as determined for routes within each topology, for other protocols such as BGP and LDP and for inter-area routes. The inheritance of RAPID alternates for PIM still requires substantial investigation. Although RAPID provides alternate paths for IGP destination, these are intended for forwarding purposes only; the alternates are not redistributed in any fashion into other protocols.

6.1 Multiple-Region Routing

The complication with inheriting alternates to routes in a different region, whether that be a different OSPF area, OSPF exterrnal routes, or a different ISIS level, is because a route in a different area may be reached via multiple border routers (area border routers (ABRs) or level boundary routers). The different scenarios and solutions will be illustrated with respect to OSPF inter-area routing.

6.1.1 OSPF Inter-Area Routes

In OSPF, each area's links are summarized into a summary LSA, which is announced into an area by an Area Border Router. ABRs announce summary LSAs into the backbone area and inject summary LSAs of the backbone area into other non-backbone areas. A route can be learned via summary LSA from one or more ABRs; such a route will be referred to as a summary route.

There are three possible scenarios which must be considered. The shortest path(s) to the summary route is learned

1. from exactly one ABR,

2. from multiple ABRs with at least two different primary neighbors,

3. from multiple ABRs with the same primary neighbor(s),

These three scenarios will be explained in reference to Fig. 25. In the first scenario, there is only a single ABR through which the summary route was learned. Therefore both the primary next-hops and alternate next-hops can be inherited from that ABR.

In the second scenario, multiple ABRs provide the shortest path to reach a summary route, but they do so via different primary neighbors. In this case, destination D can be reached via ABR1 and ABR2. ABR1 is reached via primary neighbor N1 while ABR2 is reached via primary neighbor N2. In this case, the primary neighbor for ABR1 can be used as an alternate for the primary next-hops to reach ABR2 and vice versa.

In the third scenario, although multiple ABRs provide the shortest paths to reach the destination, all the primary next-hops are via the same primary neighbor. The alternate selected must not loop traffic back to S.

First, consider the case where ABR1 has a loop-free alternate A1. The first possibility is that the set of ABRs, used by S to reach D, is equidistant from A1; if so, then given that A1 was loop-free for ABR1, A1 is also loop-free for the other ABRs in the set. The second possibility is that a subset of the ABRs is farther than ABR1; these will not be forwarded to, but the traffic won't be looped back. The third possibility is that a subset of the ABRs are closer than ABR1 for A1. This means that A1 may not be loop-free with respect to that subset of ABRs which are closer than ABR1. An alternate A1 can only be selected as an alternate for the summary route, if A1 is loop-free with respect to all the ABRs in the set. Given that the set of relevant ABRs is indexed from 1 to T, this means that the loop-free check becomes:

$$\text{Minforall } t \text{ in } T\ \ (D!S(Ni, ABRt)) - Dopt(Ni, S) < Dopt(S, ABRt)$$

Equation 9: Loop-Free check for ABRs

For U-Turn alternates, it is also a bit more complicated and a U-turn alternate must meet the following requirement.

Minforall t in T (Minforall j in J (D!S(Ri,j, ABRt) - Dopt(Ri,j, S))) < Dopt(S, ABR1)

Equation 10: U-Turn Alternate Check for ABRs

The "most loop-free" alternate will be taken.

Because the routes are summarized, any non-local SRLG information is not available.

6.1.2 OSPF External Routing

Rules of inheritance of alternate next-hops for external routes is the same as for inter-area destinations. The additional complication comes from forwarding addresses, where an ASBR uses a forwarding address to indicate to all routers in the Autonomous System to use the specified address instead of going through the ASBR. When a forwarding address has been indicated, all routers in the topology calculate the shortest path to the link specified in the external LSA. In this case, the alternate next-hop of the forwarding link should be used, in conjunction with the primary next-hop of the forwarding link, instead of those associated with the ASBR.

6.1.3 ISIS Multi-Level Routing

ISIS maintains separate databases for each level with which it is dealing. Nodes in one level do not have any information about state of nodes and edges of the other level. ISIS level boundary points , also known as ISIS level boundary routers, are attached to both levels. ISIS level boundary routers summarize the destinations in each, level. ISIS inter-level route computation is very similar to OSPF inter area routing. Rules for alternate next-hop inheritance is the same as described for OSPF inter area routing in Section 6.1.1

6.2 OSPF Virtual Links

OSPF virtual links are used to connect two disjoint backbone areas using a transit area. A virtual link is configured at the border routers of the disjoint area. There are two scenarios, depending upon the position of the root, router S.

If router S is itself an ABR or one of the endpoints of the disjoint area, then router S must resolve its paths to the destination on the other side of the disjoint area by using teh summary links in the transit area and using the closest ABR summarizing them into the transit area. This means that the data path may diverge from the virtual neighbor's control path. An ABR's primary and alternate next-hops are calculated by RAPID on the transit area.

The primary next-hops to use are determined based upon the closest set of equidistant ABRs; the same rules described in Section 6.1.1 for inter-area destinations must be followed for OSPF virtual links to determine the alternate next-hop. The same ECMP cases apply.

If router S is not an ABR, then all the destinations on the other side of the disjoint area will inherit the virtual link's endpoint, the transit ABR. The same OSPF inter-area rules described in Section 6.1.1 must be followed here as well.

Supporting non-local SRLGs is possible because if router S is an ABR which has the link state of both the transit area and the disjoint area, then S can avoid using an alternate path which shares an SRLG with the first hop of the primary path.

6.3 BGP Next-Hop Synchronization

Typically BGP prefixes are advertised with AS exit routers router-id, and AS exit routers are reached by means of IGP routes. BGP resolves its advertised next-hop to the immediate next-hop by potential recursive lookups in the routing database. RAPID computes the alternate next-hops to the all the IGP destinations, which includes alternate next-hops to the AS exit router's router-id. BGP simply inherits the alternate next-hop from IGP. The BGP decision process is unaltered; BGP continue to use the IGP optimal distance to find the nearest exit router. MBGP routes do not need to copy the alternate next hops.

6.4 RAPID with IGP Tunnels

RAPID treats IGP tunnels the same as any other link. If router S is not an endpoint of the tunnel, then the alternate path is computed as normal; due to a lack of knowledge about the SRLGs used by the tunnel, SRLG protection is not possible. If router S is one of the end-points, then all destinations which have the tunnel as a primary next-hop must be protected via a protection scheme associated with the tunnel. Such a protection scheme might be RSVP-TE Fast-Reroute or hot standby tunnels. Because the physical interface used by the tunnel is not known to RAPID, RAPID cannot compute an alternate which is link or node protecting.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.